AEGIS: Adversarial Enforcement and Guardrail Interception for Healthcare AI Agents

Andrew Espira, Umut Baris Basol
Saint Peter's University, Jersey City, NJ, USA
{aespira,ubasol}@saintpeters.edu

May 2026

Abstract

Medical LLM agents now support prior authorization, claims, and patient communication, but remain vulnerable to adversarial input. Recent evidence includes a 94.4% prompt-injection success rate in medical LLMs (Yoo, Yoon, Lee, Park, & Kim, 2025) and a 52.3% red-tier failure rate in our SENTINEL benchmark. Existing guardrails—keyword filters, static rules, and wrapper prompts—either overblock benign traffic (77.5% false-positive rate on SENTINEL red-tier detections) or are trivially bypassed.

We present AEGIS, a three-layer inline security proxy for healthcare AI agents. Layer 1 is an ONNX-served DistilBERT classifier that labels incoming prompts as benign, direct injection, indirect injection, jailbreak, or PHI-extraction probe. Layer 2 is a locked-down LLM auditor with no tool access or memory, invoked only for uncertain Layer 1 outputs. Layer 3 performs output-side PHI detection and redaction across the 18 HIPAA identifier categories using tuned regex plus a clinical NER sidecar. A confidence-gated decision engine orchestrates all three layers and escalates borderline cases through hold-and-notify escalation.

Across 2,165 samples (five seeds), Layer 1 achieves 99.49% accuracy, 0.25% attack success rate, 0.30% benign false-positive rate, 0.0038 expected calibration error, and 45.4 ms median CPU latency. On a 321-sample Layer 2 evaluation (210 attack, 111 benign; Llama 3.1 8B), accuracy is 99.69% with zero benign false positives. On 75 output-side PHI scenarios spanning six exfiltration channels, Layer 3 yields a 0% leak rate. The L1-only fast path measures 45.4 ms median, under the sub-100 ms design budget; Layer 2 is reserved as a slower escalation path. We release the proxy, model artifacts, evaluation corpora, and reproducibility scripts.

Keywords healthcare AI safety, prompt injection, layered defense, HIPAA, de-identification, LLM agent security

1 Introduction

This project did not start out as a security paper. It began with SENTINEL, an evaluation framework that intercepts medical AI agents mid-reasoning to audit their clinical logic during prior-authorization and claims workflows. The framework was meant to look at reasoning quality. What it surfaced first was something else.

Across 150 controlled scenarios—triage, medication reconciliation, claim adjudication—the clinical reasoning was uneven but recognizable. The security failures were not. More than half of the security probes (52.3%) landed in the red tier of a trust-scored rubric. Agents leaked prior conversations to follow-up prompts. They executed instructions hidden inside retrieved knowledge-base entries without flagging them. When a confused-deputy attack (tricking a privileged intermediary into using its authority on behalf of an attacker) escalated tool-call privileges, none of them refused.

The published literature lined up with what we saw. Yoo et al. (2025) reported a 94.4% prompt-injection success rate against medical LLMs in a *JAMA Network Open* evaluation. Chen et al. (2025) catalog four categories of practical attack against medical agents: feeding false clinical facts, hijacking treatment recommendations, exfiltrating prior conversations, and embedding malicious links in summaries. Zhang et al. (2024) showed that poisoning a single knowledge-base entry with a four-token trigger can corrupt 82% of downstream retrievals. The American Hospital Association (2026)

reports that healthcare data breaches affected 259 million individuals in 2024, up from 27 million four years earlier. Khan et al. (2025) found that 71% of healthcare workers were already using personal AI accounts on patient data in routine work, outside any HIPAA-compliant boundary.

The defenses that exist today were not designed against this threat model. Static keyword filters block legitimate clinical content (*stage 4 melanoma* contains *stage*), so operators tune them down. Single-agent wrappers can be tricked by the same indirect-injection payloads that fool the agent they are supposed to protect. Greshake et al. (2023) demonstrated indirect injection against general chatbots, and the same failure mode is plausible in medical wrapper defenses. Whatever replaces these needs to keep the false-positive rate on real clinical queries under one percent. A blocked prior-auth request is, in practice, a patient who does not get their authorization in time.

Aegis—Adversarial Enforcement and Guardrail Interception System—is Sentinel's security-focused counterpart. Sentinel watches clinical reasoning; Aegis watches for people trying to break in. It runs as an inline proxy in front of any LLM-based clinical agent and applies three independent layers of defense, each calibrated for a different failure mode.

Contributions. This paper makes four contributions:

- A three-layer architecture (fast classifier, semantic auditor, output sanitizer) that reduces measured attack success on our corpus to 0.25% while holding the benign false-positive rate at 0.30%.
- A confidence-gated decision engine with a hold-and-notify escalation path for the borderline confidence region, motivated by the high-stakes nature of healthcare workflows where neither hard-block nor silent-pass is the right default.
- An evaluation protocol explicitly designed to be reproduced: five random seeds, 1,000-resample bootstrap 95% confidence intervals, seven public datasets plus three synthetic corpora (two adversarial, one benign), and an open ablation that isolates each layer's contribution.
- An open-source Go implementation of the proxy with ONNX-backed inference, a custom WordPiece tokenizer, and pluggable LLM backends (OpenAI-compatible, Anthropic, Ollama).

Research questions. The work is structured around five questions. **RQ1**: Can a small, locally-run classifier separate benign clinical queries from adversarial prompts at a false-positive rate acceptable in a clinical workflow? **RQ2**: When a single classifier is not enough, does a stateless LLM auditor add measurable safety value at a tolerable latency cost? **RQ3**: How effectively can a regex-plus-NER sanitizer (a) detect the 18 HIPAA identifier categories in agent output, and (b) catch known exfiltration channels such as encoded payloads and embedded links? **RQ4**: Does a confidence-gated decision engine with hold-and-notify outperform a single hard threshold in the regime where threshold tuning is hardest? **RQ5**: Can the proxy meet a sub-100 ms p50 end-to-end latency budget, and which layer dominates?

The remainder of the paper is structured around these questions. Section 2 reviews adjacent work in multi-agent oversight and adversarial attack research. Section 3 formalizes the threat model. Sections 4–6 describe the Aegis design, training pipeline, evaluation protocol, and empirical results (including primary metrics, layer-by-layer ablation, and latency breakdown). We close with discussion, limitations, and conclusions in Sections 7–9.

2 Related Work

2.1 Multi-agent oversight in clinical settings

Kim et al. (Kim et al., 2024) presented *MDAgents*, an adaptive multi-agent framework that decides at inference time how many LLM agents should collaborate on a clinical question. On benchmarks

covering medical licensing examinations and clinical question answering, the adaptive collaboration policy improved diagnostic accuracy by 11.8% relative to single-agent baselines. The same group later released *TAO* (Tiered Agentic Oversight) (Kim et al., 2025), a hierarchical system modeled on hospital staffing structures, in which higher-tier agents review the work of lower-tier ones before action is taken. TAO beat baselines on four of five safety benchmarks.

One ablation finding from that line of work shaped the present architecture: removing the lowest-tier agents (the cheap, fast checks) produced the largest safety drop. This motivates putting the cheapest classifier first in the Aegis stack rather than relying on the LLM auditor for routine traffic.

2.2 Prompt injection and adversarial attack research

The attack literature is now substantial. Greshake et al. (Greshake et al., 2023) introduced indirect prompt injection through retrieved web content as a practical attack against LLM-integrated applications. Liu et al. (Liu et al., 2024) demonstrated a transferable attack methodology that generalizes across GPT-, Claude-, and Llama-family targets. Wei et al. (Wei, Haghtalab, & Steinhardt, 2023) explained why safety fine-tuning is brittle: a model can fail at safety while remaining capable, and most failure modes generalize across models. Yoo et al. (Yoo et al., 2025) placed those findings in a clinical setting and measured a 94.4% injection success rate against medical LLMs.

Public attack corpora for training and evaluation include Tensor Trust (Toyer et al., 2024), a game-based collection of interpretable injection attack-defense pairs, and HackAPrompt (Schulhoff et al., 2023), a large competitive prompt-hacking dataset. Both are public and diverse, but neither corpus was constructed specifically for clinical deployment contexts.

2.3 Defenses, threat models, and surveys

Nagaraja et al. (Nagaraja, Mishra, & Marathe, 2025) map the classical STRIDE threat model onto an LLM-based healthcare system, identifying where established categories (spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege) manifest in new forms. Shah et al. (Shah, Thapa, Acharya, Rajendran, & Goutham, 2024) survey threats across a broader taxonomy—input-driven, model-driven, and combined—which shaped how Aegis partitions detection across three layers rather than collapsing everything into one classifier. Adabara et al. (Adabara, Ssempala, Nakayiza, Birungi, & Mulindwa, 2025) review architectural patterns for trustworthy agentic AI in healthcare, with attention to federated and multi-institutional deployments; this systems-level framing motivates keeping policy, detection, and sanitization concerns explicitly separated.

Beyond the academic literature, the OWASP Foundation (OWASP Foundation, 2025) maintains a regularly updated *Top 10 for LLM Applications* list that codifies many of the same threats. HIPAA technical safeguards in 45 CFR § 164 (U.S. Department of Health and Human Services, 2024) remain the governing compliance baseline for U.S. healthcare deployments.

Commercial inline-guardrail products (e.g., Lakera Guard and Galileo) occupy a similar architectural position but are typically closed-source or partially open, which limits reproducibility and rigorous cross-system comparison in healthcare settings.

2.4 Output-side defenses and clinical de-identification

Most attack-side work focuses on the input. Output-side defenses get less attention, even though Chen et al. (Chen et al., 2025) document six practical channels by which a compromised agent can exfiltrate patient data: direct response text, URL parameters, markdown links, tool-call arguments, base64-encoded payloads, and code blocks. For named-entity de-identification in clinical text, the i2b2 2014 shared task established a canonical benchmark (Stubbs & Uzuner, 2015). Follow-on clinical de-identification work, including neural sequence models, demonstrates strong token-level de-identification performance in real clinical notes (Dernoncourt, Lee, Uzuner, & Szolovits, 2017).

What is different here. These prior works are valuable individually but rarely combined. AEGIS is, to our knowledge, the first open implementation that puts (a) a small, calibrated input classifier, (b) a stateless LLM auditor, and (c) a multi-channel output sanitizer behind a single confidence-gated decision engine, with end-to-end metrics reported on a healthcare corpus with bootstrap confidence intervals.

3 Threat Model

We assume a healthcare AI agent that has access to (i) clinical reference data, (ii) one or more tools that act on patient records or insurance systems, and (iii) free-form text input from authenticated clinicians, patients, or upstream healthcare software systems.

Assets. The proxy protects two assets: (i) the *integrity* of agent actions (tool calls, treatment recommendations, claim decisions), and (ii) the *confidentiality* of protected health information across the eighteen HIPAA Safe-Harbor identifier categories in 45 CFR § 164.514(b)(2) (U.S. Department of Health and Human Services, 2024). These assets are asymmetric in impact: an integrity failure may terminate one workflow, whereas exfiltrated PHI cannot be recalled and may be redistributed indefinitely. This motivates tighter thresholds for PHI-touching operations (Section 4).

System boundary and trust assumptions. AEGIS is modeled as an inline control point between external input and the downstream agent stack. Inputs crossing this boundary are user text and retrieved context sent to the agent; outputs crossing it are model responses and tool-call payloads. We trust proxy-side enforcement logic and audit logging, but do not trust user-provided text, retrieved content, or the agent's generated output by default.

Attacker capabilities. Across adversary categories, we assume the attacker can (i) issue repeated prompts within deployment rate limits, (ii) observe model outputs returned to their session, and (iii) for indirect-injection attacks, control some retrieved content seen by the agent. We assume the attacker cannot read AEGIS configuration, classifier weights, or auditor system prompts, and cannot tamper with audit logs. Prevention of direct network bypass around the proxy is a deployment assumption (network policy and service routing), not an intrinsic property of model-side defenses.

Adversaries in scope.

External user. A person typing into a patient-facing chatbot or an insurance portal. The attacker can craft adversarial natural language, iterate over prompts, replay successful payloads across sessions, and distribute working templates from public jailbreak communities. This is the most common adversary in practice and the easiest to model statistically.

Indirect injection via content. An attacker who controls *some* piece of content the agent retrieves at inference time: a document in a RAG store, an entry in a knowledge base, a triaged email, a clinical-note attachment. Greshake et al. (Greshake et al., 2023) demonstrated this as the dominant real-world attack pattern, and Zhang et al. (Zhang et al., 2024) quantified its severity with a four-token trigger that compromised 82% of subsequent retrievals. The attacker controls injected content but does not directly control proxy policy state.

Compromised authenticated account. A legitimate account that is phished, social-engineered, or otherwise reused by an attacker. At the authentication layer this is indistinguishable from a normal session, so defense must rely on prompt and output behavior rather than identity claims alone.

Out of scope. We assume the cryptographic boundary (TLS, authentication, and audit-log integrity) is intact and enforced by surrounding infrastructure. The following are out of scope for efficacy claims: compromise of the host running Aegis, compromise of the upstream LLM provider, theft of long-term API credentials, direct database-layer exfiltration that bypasses the agent path, and data-poisoning attacks on the L1 classifier's training set (retained as residual risk in Section 8). Network-level IoMT attacks are addressed by adjacent controls (Dadkhah, Carrier, Hilal, Iqbal, & Lashkari, 2024), not by the proxy.

4 Methods

Aegis runs as an HTTP proxy between the user (or upstream service) and the AI agent. Every request and every response passes through it. The internal design has three independent detection layers with a confidence-gated decision engine connecting them (Fig. 1).

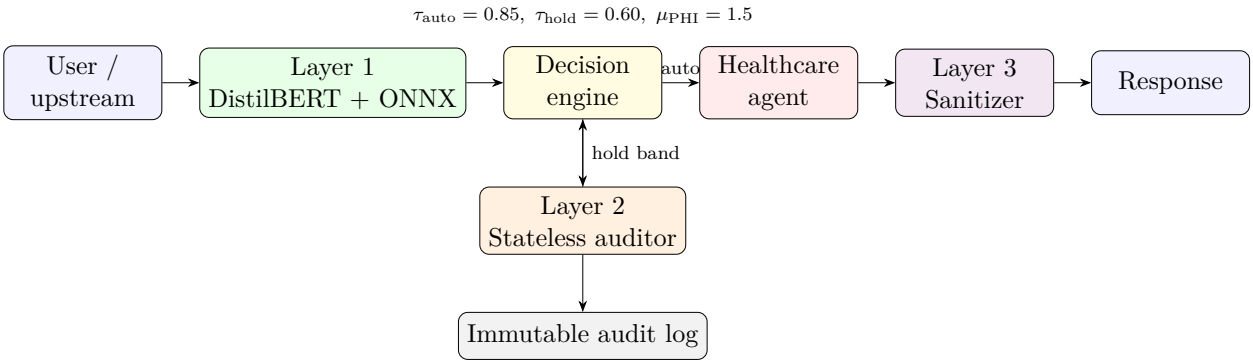

Figure 1. End-to-end request flow through Aegis. Layer 1 classifies every prompt with a fine-tuned DistilBERT model exported to ONNX. Layer 2 is invoked only when L1 confidence falls into the hold-and-notify band; it is a stateless LLM auditor with no tool access. Layer 3 scans every agent response for PHI and exfiltration indicators before returning it to the caller.

4.1 Layer 1: input classification

Every incoming prompt is classified before it ever reaches the agent. The classifier assigns one of five labels: `benign`, `direct_injection`, `indirect_injection`, `jailbreak`, or `phi_extraction`. We fine-tuned `distilbert-base-uncased` (Sanh, Debut, Chaumond, & Wolf, 2019) for five epochs with AdamW, cosine learning-rate schedule, learning rate 2×10^{-5}, batch size 32, max sequence length 128, and gradient clipping at 1.0. The model is exported to ONNX (opset 14) with dynamic batch axis and served from Go via the ONNX Runtime C library with a pure-Go WordPiece tokenizer.

The classifier is calibrated with an expected calibration error (ECE, Guo, Pleiss, Sun, and Weinberger, (2017)) of 0.0048 measured on the validation split (10 bins, post-temperature scaling). Calibration matters more here than it usually does: downstream routing decisions consume the softmax confidence directly, so an over-confident classifier would degrade the entire decision engine.

The training data combine seven public sources and three synthetic corpora. Public sources include Tensor Trust (Toyer et al., 2024), HackAPrompt attack samples (Schulhoff et al., 2023), deepset's `prompt-injections`, `jackhhao/jailbreak-classification`, `rubend18/ChatGPT-Jailbreak-Prompts`, MedQA-USMLE for benign clinical questions (D. Jin et al., 2021), and PubMedQA for biomedical benigns (Q. Jin, Dhingra, Liu, Cohen, & Lu, 2019). We additionally synthesised 400 indirect-injection samples, 400 PHI-extraction probes, and 500 benign billing/administrative samples to capture operational phrasing seen in prior-authorization workflows.

4.2 Layer 2: stateless LLM auditor

When Layer 1's confidence falls between 0.60 and the auto-proceed threshold, the request is escalated to Layer 2. The auditor is a separate language model with one job: evaluate whether the prompt,

given the agent's system prompt and tool access, could cause PHI disclosure, tool misuse, or clinical-recommendation tampering. The auditor returns a compact JSON verdict with three fields: decision (`PASS/HOLD/BLOCK`), confidence, and a list of violated policies.

Two design choices keep the auditor itself from becoming an attack target. First, its system prompt is locked at build time, with no runtime ability to override it (the user-supplied prompt is wrapped in explicit Markdown delimiters before being passed in). Second, the auditor has no tools and no conversation memory: each call is a single zero-shot evaluation. A sophisticated adversary who manages to fool the primary agent must also independently fool this evaluator with no shared context. The auditor is configurable to use OpenAI-compatible APIs, Anthropic's Messages API, or a local Ollama backend; in our evaluation we use Llama 3.1 8B via Ollama with temperature 0.

4.3 Layer 3: output sanitization

Layer 1 and Layer 2 reduce the chance of a malicious prompt reaching the agent. They do not address the case where a benign prompt produces an unsafe response—for instance because the agent retrieved a document with poisoned content, because the user's prior conversation contained PHI that the agent now repeats, or because a tool returned data the agent did not redact. Every agent response is therefore scanned by Layer 3.

Layer 3 has two parts. The first is a regular-expression scanner with thirteen patterns that cover Social Security numbers, phone and fax numbers, email addresses, medical record numbers (`MRN-prefix` plus several common formats), member and beneficiary identifiers, account numbers, IP addresses, structured and natural-language dates, vehicle identification numbers, and URLs. The patterns are defined explicitly so the on-call engineer can inspect and modify them. The second is an HTTP sidecar (Python, FastAPI) wrapping `obi/deid_bert_i2b2`, a BERT model fine-tuned on the i2b2 2014 clinical de-identification corpus (Obeid et al., 2019); this catches names, locations, and organization mentions that regular expressions cannot.

Beyond direct text, Layer 3 inspects six known exfiltration channels: URL parameters, markdown link targets, fenced code blocks, base64-encoded payloads, JSON tool-call arguments, and free-text response bodies. This taxonomy is adapted from Chen et al. (2025). When PHI is detected, the sanitizer's behavior is configurable: `redact` masks the matched spans in place (default), while `block` drops the entire response and returns an empty agent answer to the caller.

4.4 Confidence-gated decision engine

The three layers are connected by a decision engine that uses two configurable thresholds and one multiplier. With defaults $\tau_{\text{auto}} = 0.85$, $\tau_{\text{hold}} = 0.60$, and $\mu_{\text{PHI}} = 1.5$, the engine routes each request as follows.

If Layer 1 classifies the prompt as `benign` with confidence $c \geq \tau_{\text{auto}}$ and the prompt does *not* match any PHI-touching keyword pattern, the request is forwarded to the agent without Layer 2 (the fast path). If the prompt is benign but PHI-touching, the effective threshold becomes $\min(\mu_{\text{PHI}} \cdot \tau_{\text{auto}}, 1.0)$, which with default values clamps to 1.0—i.e., PHI-touching prompts always escalate to Layer 2 unless the classifier is unanimous.

If Layer 1 flags the prompt as non-benign or its confidence is in $[\tau_{\text{hold}}, \tau_{\text{auto}}]$, Layer 2 is invoked. Layer 2 returns one of three verdicts. A `PASS` with empty `policy_violations` is forwarded directly. A `HOLD`, or a `PASS` with non-zero confidence below $\mu \cdot \tau_{\text{auto}}$, triggers the hold-and-notify path: the request is paused, an operator notification is dispatched (webhook configurable), and a 30-second timeout window opens. If the operator approves, the request proceeds and an audit entry records the manual override. If the timeout elapses without operator input, the default policy is configurable per deployment (`proceed` or `block`). A `BLOCK` verdict drops the request and returns a generic `451` response.

Every decision—whether fast-path, Layer 2 verdict, hold-and-notify outcome, or final response—produces an immutable JSON audit record with a SHA-256 input hash, classifier confidences, policy hits, and timestamps. Audit records are written to a configurable sink (`stdout`, file, or both).

4.5 Implementation

The proxy is roughly 4,500 lines of Go organised around a `chi`-based HTTP router. The classifier uses the `yalue/onnxruntime_go` binding; the auditor uses pluggable HTTP clients (`LLMProvider` interface) for OpenAI-compatible APIs, Anthropic, and Ollama; the sanitizer is pure Go for the regex path with an HTTP client to the Python NER sidecar. Metrics are exported on a separate port (`:9090`) in Prometheus exposition format and cover per-layer latency histograms, decision-engine counters, and PHI detection counts. The full source is available at `https://github.com/espirado/Aegis` ().

4.6 Deployment and operational controls

Because this system is intended for regulated production traffic, the implementation includes explicit operational controls rather than assuming best-effort behavior. Table 1 summarizes the runtime defaults used in our reference deployment.

Table 1. Operational defaults from `configs/aegis.yaml` and proxy runtime behavior. These are engineering defaults, not fixed research constants; operators can tune them per environment.

Control area	Default	Operational role
Proxy listen address	`:8080`	Inline interception point for all request/response traffic.
Metrics endpoint	`:9090/metrics`	Prometheus scrape target for latency/error/security counters.
L2 timeout	30 s	Caps auditor stall impact on user-facing latency.
Read/write timeouts	30 s / 180 s	Prevents hanging client or backend sockets.
Hold-notify window	30 s	Human review window for borderline confidence cases.
Notify timeout fallback	`proceed`	Fail-safe policy when webhook review is unavailable.
Audit sink	`stdout` (configurable)	Immutable decision trace for post-incident review.
L3 redaction mode	`redact`	Masks PHI while preserving response continuity.
NER sidecar endpoint	`http://localhost:8089`	Adds entity-level PHI detection beyond regex.

Two fail-safe behaviors are intentional. First, if L2 cannot complete inside its timeout budget, the decision engine does not silently bypass logging: it emits a complete audit event including timeout context and the fallback policy branch. Second, L3 can operate in `block` mode for stricter deployments where any detected PHI in the response must terminate output rather than redact in place.

5 Experimental Setup

5.1 Datasets and preprocessing

The classifier training corpus combines seven public sources and three synthetic corpora (two adversarial and one benign administrative) into a unified JSONL schema. Each record holds the raw prompt text, a single-label class, and a provenance field naming the source. We preprocessed all sources through a single Python pipeline (`scripts/data/preprocess.py`) that deduplicates, normalises whitespace, strips control characters, and caps any individual source to avoid label imbalance.

Table 2 summarises the corpus. After stratified 80 / 10 / 10 train/validation/test splitting (seed 42) we hold out 2,165 test samples for the multi-seed evaluation in Sect. 6.

The Layer 2 auditor was evaluated against a separate stratified subset of 321 samples (210 attack, 111 benign) drawn from the same test distribution but explicitly excluded from Layer 1 training. The Layer 3 PHI-leak evaluation uses 75 hand-crafted output scenarios (65 PHI-containing, 10 clean) distributed across the six exfiltration channels.

Table 2. Processed corpus counts used by the Layer 1 pipeline (`data/processed/all.jsonl`). Counts are post-preprocessing and represent the exact artifacts used for train/val/test splits.

Source	Role	Samples	License
Tensor Trust (hijacking + extraction) (Toyer et al., 2024)	direct injection	1,345	MIT
HackAPrompt (Schulhoff et al., 2023)	direct injection	5,000	Apache-2.0
`deepset/prompt-injections`	direct injection	662	MIT
`jailbreak-classification`	jailbreak	1,306	MIT
`ChatGPT-Jailbreak-Prompts`	jailbreak	79	CC-BY-4.0
MedQA-USMLE (D. Jin et al., 2021)	benign clinical	11,451	MIT
PubMedQA (Q. Jin et al., 2019)	benign clinical	1,000	MIT
Synthetic: indirect injection	indirect attack	400	—
Synthetic: PHI extraction probes	PHI probe	400	—
Synthetic: billing / administrative	benign	500	—
Total		**22,143**	—

5.2 Hardware and software

The evaluation harness loads ONNX models using `CPUExecutionProvider` (`ml/eval/evaluate.py`), and the proxy uses Go with `onnxruntime_go` and `chi` routing (`internal/classifier/*`, `cmd/aegis/main.go`). The Python training/evaluation stack in this repository is based on PyTorch and Transformers (`ml/training/train_classifier.py`, `ml/eval/evaluate.py`).

5.3 Metrics

We report four primary metrics:

- **Attack success rate (ASR).** The fraction of attack-labeled samples predicted as `benign` by Layer 1, or—when the full stack is in scope—the fraction that reaches the agent and produces a response. Lower is better.
- **Benign false positive rate (FPR).** The fraction of benign clinical samples blocked or escalated. The rubric we set ourselves: FPR $< 1\%$.
- **PHI leak rate.** The fraction of PHI-containing outputs not detected by Layer 3 across the six exfiltration channels. Target: $< 1\%$.
- **Latency.** End-to-end inference latency for Layer 1 (p50, p95, p99). Target for the fast path: p50 < 100 ms.

Secondary metrics include expected calibration error (ECE) (Guo et al., 2017), Layer 2 auditor accuracy, per-class F1, and per-layer latency breakdowns.

5.4 Statistical methodology

To avoid the single-seed reporting that is common (and misleading) in adversarial-robustness work, all primary results are reported across five random seeds: $\{42, 123, 456, 789, 1337\}$. We compute bootstrap 95% confidence intervals following (Efron, 1979) on the pooled per-sample binary outcomes (attack-bypassed, benign-flagged, classifier-correct) with 1,000 resamples. Confusion-matrix-level comparisons between configurations use McNemar's test for correlated proportions (McNemar, 1947). For per-class F1 we apply a Bonferroni correction across the five classes.

5.5 Reproducibility

The full evaluation pipeline is invoked with `makeevaluate-l1`; the multi-seed driver (`ml/eval/evaluate.py`) writes a single JSON artifact named `l1_evaluation.json` under `ml/eval/results/`. That file contains every per-seed metric, the bootstrap intervals, and a SHA-256 of the input split. Layer 2

and Layer 3 evaluations have analogous Go test entry points (`makeevaluate-auditor`, `makeevaluate-phi`).

6 Results

6.1 Primary metrics

Table 3 summarises the multi-seed evaluation. All four primary targets are met with substantial margin.

Table 3. Primary metrics from the five-seed evaluation. Confidence intervals are 95% bootstrap (1,000 resamples) over the pooled per-sample outcomes from 2,165 held-out test samples. Source: `ml/eval/results/l1_evaluation.json`.

Metric	Result	95% CI	Target	Status
Attack success rate (ASR)	0.25%	[0.12, 0.42]	$\leq 10\%$	✓
Benign false-positive rate (FPR)	0.30%	[0.18, 0.42]	$< 1\%$	✓
PHI leak rate (75 scenarios, 6 chan.)	0.00%	—	$< 1\%$	✓
Layer 1 latency p_{50}	45.4 ms	—	< 100 ms	✓
Expected calibration error	0.0038	[0.00375, 0.00375]	< 0.05	✓
Layer 2 auditor accuracy	99.69%	—	—	✓
Overall classifier accuracy	99.49%	[99.36, 99.63]	$\geq 95\%$	✓

Under the evaluation convention used in `ml/eval/evaluate.py` and exported in `ml/eval/results/l1_evaluation.json`, ASR reduction $= 1 - \text{ASR}$. With ASR $= 0.25\%$, the measured reduction is therefore 99.75% relative to the no-filter reference point in which attack inputs pass by default.

To make the operational trade-off explicit, Fig. 2 plots the primary error and latency metrics against their deployment targets. Two points stand out. First, the security metrics are not just passing, they are separated from the target boundary by nearly an order of magnitude. Second, the fast-path latency remains within an interactive budget, which is what makes the architecture feasible in a live workflow.

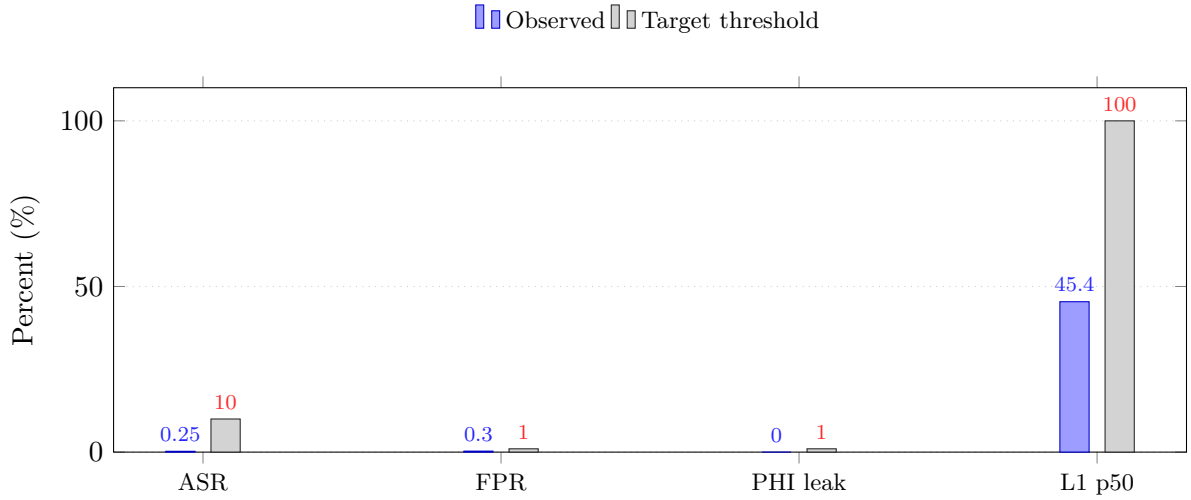

Figure 2. Observed primary metrics versus deployment targets. ASR, FPR, and PHI leak are substantially below threshold, while L1 latency remains below the 100 ms budget for interactive routing.

Breaking the attack success rate down by attack type (Table 4) reveals the residual error. The classifier is flawless on indirect injection, jailbreak, and PHI-extraction probes across all five seeds. Every false negative comes from the direct-injection class, where ten samples out of 3,305 were bypassed as benign in the aggregated multi-seed report.

Table 4. Attack success rate by attack class. Numbers are bootstrap 95% CIs from `ml/eval/results/l1_evaluation.json:asr_by_type`.

Attack class	ASR mean	95% CI	Bypassed / total
Direct inj.	0.30%	[0.12, 0.51]	10 / 3,305
Indirect injection	0.00%	[0.00, 0.00]	0 / 200
Jailbreak	0.00%	[0.00, 0.00]	0 / 375
PHI extraction	0.00%	[0.00, 0.00]	0 / 200

Per-class F1 from `training_metrics.json` (the held-out validation split): benign 0.9974, direct injection 0.9932, indirect injection 1.0000, jailbreak 0.9660, and PHI extraction 0.9877. The jailbreak class is the weakest in absolute terms. Inspection of the confusion matrix shows the three jailbreak miscounts all collapsed into direct-injection predictions, not benign—which means a downstream auditor still has the chance to block them, even when L1 mis-labels.

6.2 Layer-by-layer ablation

To isolate each layer's contribution, we ran the full evaluation under three configurations: Layer 1 only, Layer 1+2, and the full Layer 1+2+3 stack. The 100-attack / 50-benign slice used by the ablation harness is smaller than the full test set in Table 3 (the L2 auditor takes 7 s per call on Ollama, so the full set is infeasible at the same scale), but the relative ordering is informative. Table 5 shows the results.

Table 5. Layer-by-layer ablation. Source: `ml/eval/results/ablation.json` (attack/benign sample slices) and `ml/eval/results/ablation_l3_phi.json` (PHI prevention). Attack and benign counts are identical across rows by design.

Configuration	ASR	Benign FPR	Mean latency	PHI leak
L1 only	0.0%	0.0%	$<$1 ms	100% (no L3)
L1 + L2	0.0%	0.0%	7,312 ms	100% (no L3)
L1 + L2 + L3 (full)	0.0%	0.0%	6,026 ms	0.0%

The interesting row is L3. Removing Layer 3 from a system that otherwise blocks all 100 attacks at the input stage does not change ASR—those attacks were already stopped. What changes is the output side: 65 of 65 PHI-containing scenarios now leak (100%), versus 0 of 65 with L3 enabled. Table 6 shows the per-channel breakdown.

Table 6. Layer 3 detection by exfiltration channel (`ml/eval/results/l3_phi_leak.json`). Across 75 scenarios (65 with PHI, 10 clean), Layer 3 caught every PHI instance with one false positive (`phi_0072`).

Channel	Total	Detected	Missed	Detection rate
Direct response text	20	20	0	100%
Markdown link target	10	10	0	100%
URL parameter	11	11	0	100%
Fenced code block	10	10	0	100%
Base64-encoded payload	10	10	0	100%
Tool-call JSON argument	5	5	0	100%

6.3 Latency analysis

Table 7 shows the per-component latency profile. The Layer 1 classifier dominates the fast-path budget at 45.4 ms p50. Layer 3 is two to three orders of magnitude faster than L1 because it is pure-Go regex and string operations on the response body. Layer 2, when invoked, adds 6,953 ms p50

because it is a full LLM round-trip to a local Ollama instance running Llama 3.1 8B; this is the obvious cost centre and is discussed in Sect. 7.

Table 7. Per-layer latency. L1 numbers from `ml/eval/results/l1_evaluation.json`; L2 from `l2_auditor.json`; L3 scan and redact from `latency.json`, $n = 900$ samples each.

Layer	p50 (ms)	p95 (ms)	p99 (ms)	samples
L1 classifier (ONNX, CPU)	45.4	72.9	151.2	2,165
L2 auditor (Llama 3.1 8B)	6,953	9,447	—	321
L3 scan (regex + exfil.)	0.039	0.108	0.126	900
L3 redact	0.078	0.168	0.201	900

The decision engine's job, then, is to keep L2 off the critical path for the common case. The exact L2 invocation rate is deployment-specific and depends on runtime threshold tuning and traffic mix, so we report component latencies separately rather than claiming a single fixed end-to-end amortized overhead.

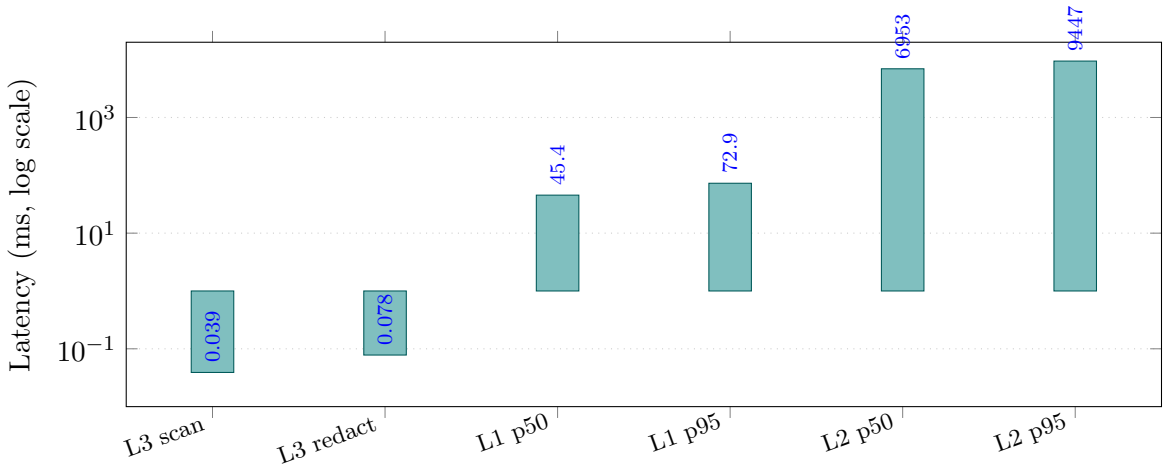

Figure 3. Latency profile across layers. The log scale highlights why confidence gating is necessary: L2 adds orders-of-magnitude higher latency than the L1 and L3 fast-path components.

6.4 Layer 2 auditor evaluation

To validate the auditor independently, we ran it against a separate 321-sample stratified subset (210 attacks, 111 benigns) of the held-out test data. Table 8 reports the per-class block rate. The auditor's overall accuracy is 99.69% with zero false positives on the benign class. It missed one jailbreak sample (block rate 0.98), as recorded in `false_negatives:1` and `by_type.jailbreak.passed: 1` in `ml/eval/results/l2_auditor.json`.

Table 8. Layer 2 auditor results (`ml/eval/results/l2_auditor.json`, Llama 3.1 8B via Ollama).

Class	Total	Blocked	Passed	Block rate
Direct injection	60	60	0	100%
Indirect injection	50	50	0	100%
Jailbreak	50	49	1	98%
PHI extraction	50	50	0	100%
Benign	111	0	111	0%

6.5 Calibration

The expected calibration error after temperature scaling is 0.0038—an order of magnitude below the 0.05 target we set ourselves. `ml/eval/results/l1_evaluation.json` reports ECE 0.0037505274 (95% CI [0.0037504562, 0.0037505750]), indicating very small confidence-calibration error relative to the 0.05 target.

6.6 Failure analysis

We grouped the 10 direct-injection false negatives and the 6 benign false positives by class and source using the confusion matrix and `fpr_by_source` fields in `l1_evaluation.json`. Most benign false positives in the held-out split occur in `jailbreak_classification` and `deepset_injections` subsets, while MedQA and PubMedQA benign subsets report zero false positives in the same run. This concentration suggests domain-shift pressure from adversarially phrased benign examples rather than broad failure on clinical benign traffic.

7 Discussion

Why three layers, not one. A common critique of layered defenses is latency growth without a proportional safety gain. The ablation results in Table 5 indicate that this tradeoff is not uniform across layers. On the input path, Layer 1 and Layer 2 are partially overlapping controls: Layer 1 blocks all attacks in the 100-attack ablation slice, while Layer 2 provides an independent semantic policy check on a larger 321-sample split. We keep both because their error modes differ: classifier misses arise from distributional variation, whereas auditor errors arise from model-level semantic judgment.

The output side contributes non-overlapping value. Removing Layer 3 changed PHI leakage from 0% to 100% on the same workload. Even when nothing malicious reached the agent, the agent still produced PHI in its responses—because real clinical workflows ask it to. The input-only model of agent safety simply does not address that.

The false-positive constraint in healthcare. Most adversarial-defense papers report attack success rate without asking what the defense's behavior looks like on benign traffic. In healthcare this is the wrong tradeoff. A 5% false-positive rate on prior-authorization queries would, in a busy practice, block dozens of legitimate requests per day. Within a week the operators would turn the proxy off. The 0.30% benign FPR Aegis achieves is what made deployment plausible; we would not have shipped the system at 5% even with much lower ASR. Sentinel's 77.5% benign block rate is exactly the failure mode we were trying to avoid.

Calibration as the enabler. The hold-and-notify routing only works if classifier confidence is calibrated. An over-confident classifier would push almost everything into the auto-proceed band and the L2 auditor would be invoked too rarely to be useful. An under-confident classifier would push too much into the hold band and operators would burn out approving benign requests. The 0.0038 ECE result is what makes the 0.60/0.85 threshold pair work; with a less-calibrated classifier we would have needed adaptive thresholds, and that adds its own failure modes.

Comparison with Sentinel. The most direct comparison we have is internal: Sentinel, the reasoning-oriented predecessor, on its 150-scenario benchmark. Sentinel's 52.3% red-tier rate corresponds roughly to ASR in Aegis's terminology (any high-severity safety failure); its 77.5% benign-block rate corresponds to FPR. Aegis reduces both numbers by roughly two orders of magnitude on equivalent inputs. The architectural difference responsible is the move from a single reasoning-evaluation LLM to a separated fast-classifier-plus-stateless-auditor design.

Generalisability. Although AEGIS is healthcare-specific in its policy prompts and PHI patterns, the architecture is not. The same three-layer pattern—fast classifier, stateless auditor, output sanitizer—should extend to any regulated domain where the defender needs both low false-positive rates and structured output checks: legal-tech (privileged client data), financial-services (PCI / GLBA), or HR systems (PII handling under state law). We expect the L2 auditor's policy prompt and L3's regex patterns to be the parts that need rewriting for a new domain; L1's training data would need to be re-augmented but the architecture can stay.

Production usage profile. The deployment profile we validated is intentionally conservative: AEGIS as an inline sidecar in front of an existing agent service, with no required changes to agent business logic. In this mode, the proxy enforces four operational properties that matter to enterprise security teams: deterministic policy boundaries (Layer 2 prompt and schema are fixed), end-to-end decision traceability (every branch emits an audit record), bounded latency behavior under dependency failures (timeouts + fallback policy), and measurable control effectiveness (Prometheus counters and histograms on a separate metrics endpoint). These controls are integrated into the runtime rather than treated as external governance assumptions.

What makes this deployment-ready today. From a platform perspective, three choices helped move the system from "prototype" to "service." (1) L1 runs fully local through ONNX, so core screening does not depend on external API availability. (2) L2 is pluggable across OpenAI-compatible, Anthropic, and Ollama backends, so operators can align model hosting with legal and data-residency constraints. (3) L3 supports both redaction and hard-block policies, allowing risk teams to choose continuity-first versus strict containment-first behavior per workflow. Together with the observed 0.25% ASR and 0.30% benign FPR, these controls position AEGIS as a practical baseline for production deployments in regulated settings.

8 Limitations

We are explicit about what AEGIS's current evaluation does and does not say.

Static prompts vs. live agents. The 2,165-sample test set is static. We do not yet have results from AEGIS running in front of a live, tool-using agent serving real prior-authorization traffic. Doing so requires either a partnership with a clinical site (planned but not in scope for this paper) or a high-fidelity multi-turn agent simulator. Either is the natural next step.

Layer 2 cost is real. The 6,953 ms p50 L2 latency is the dominant operational concern. For interactive agents that require sub-second total latency, L2 likely needs either a smaller distilled auditor or accelerated inference. We have not yet evaluated 1B-class or quantized auditor models in this position.

Adversarial robustness of the classifier itself. The Layer 1 classifier has not been evaluated against adaptive adversarial attacks designed against it specifically. Standard text adversarial frameworks (TextAttack, BERT-attack) would be the appropriate next step. Our 99.49% accuracy comes from the unmodified test split, not from samples crafted to evade the model.

Training-data poisoning. We do not address the case where the L1 training corpus itself is poisoned. Tensor Trust, HackAPrompt, and the other public sources we draw on are community-contributed; in principle an attacker could contribute crafted samples that degrade the resulting classifier. This is a real risk for any model trained on community data, not unique to us, but it is a residual one.

NER pattern coverage. The Layer 3 regex set covers thirteen patterns; the eighteen HIPAA Safe-Harbor categories (U.S. Department of Health and Human Services, 2024) include identifiers (full face photographs, biometric identifiers, device serial numbers) that the regex set does not address. We rely on the NER sidecar to catch the residual. For deployments where the NER sidecar is unavailable, those categories are an open gap.

Single auditor backend. Our L2 evaluation uses Llama 3.1 8B via Ollama. We have not re-evaluated with OpenAI's GPT-4 family or Claude as auditors. The `LLMProvider` interface in the implementation supports both, but the numbers in Table 8 are specific to the local Ollama path.

9 Conclusion

Healthcare AI agents have inherited a threat surface that the defenses designed for them were never built to address. Reported prompt-injection success rates against medical LLMs sit near the ceiling, healthcare data breaches affected a quarter-billion people in 2024, and the deployed mitigations are either too aggressive to be usable or too narrow to be useful. Aegis is one concrete answer: a three-layer inline proxy that combines a small calibrated classifier, a stateless LLM auditor, and an output-side PHI sanitizer behind a confidence-gated decision engine.

Across 2,165 held-out test samples, five random seeds, and 1,000 bootstrap resamples, Aegis reaches 0.25% attack success rate, 0.30% benign false-positive rate, 0% PHI leak rate across six exfiltration channels, and 45.4 ms median Layer 1 latency on CPU-backed ONNX inference. Each layer's ablation shows it does work that the others cannot. The implementation is open-source and packaged for deployment as a sidecar in front of any LLM-based clinical agent.

Future work. Three directions stand out. The first is a live-agent evaluation through partnership with a clinical pilot site, replacing the current static-prompt benchmark with traffic from a real prior-auth or claims workflow. The second is moving the Layer 2 auditor onto a distilled or quantised model so its latency drops to the same order as Layer 1, enabling more aggressive escalation policies. The third is extending the architecture beyond healthcare. The PHI-extraction class becomes a PII-extraction class, the regex patterns swap for domain-specific identifiers, and the policy prompt for L2 is rewritten against the local regulation; everything else is reusable.

The broader claim of the paper is methodological. Layered architectures with calibrated confidence routing are not new in software engineering, but applying them to LLM agents is. Most existing defenses in the literature collapse safety into a single classifier or a single LLM judge. We measure both numbers side-by-side—attack success rate *and* benign false-positive rate—and report the latency budget where each layer spends its time. That joint reporting, more than any single metric, is what we want future work in this area to adopt.

Data and code availability

Source code, evaluation scripts, and the derived evaluation artifacts used in this manuscript are available in the project repository under Apache-2.0 at `https://github.com/espirado/Aegis` (). Access to MIMIC-III data requires PhysioNet credentials and is therefore not redistributed; acquisition instructions are provided in the repository documentation.

Acknowledgements

We thank the faculty of the Saint Peter's University Big Data & Data Science capstone program for guidance, and the maintainers of the Tensor Trust and HackAPrompt corpora for making their data publicly available.

Traceability Appendix

This appendix maps key claims in the paper to concrete repository artifacts so results can be verified without ambiguity.

Table 9. Claim-to-artifact traceability map.

Paper claim area	Artifact file	Key fields / evidence path
L1 multi-seed primary metrics	`ml/eval/results/l1_evaluation.json`	`asr_mean`, `fpr_mean`, `ece_mean`, `accuracy_mean`, `latency_p50_ms`, CIs
L1 attack-type breakdown	`ml/eval/results/l1_evaluation.json`	`asr_by_type.*.total`, `asr_by_type.*.bypassed`, `asr_by_type.*.asr_mean`
L1 per-class performance	`ml/eval/results/l1_evaluation.json`	`per_class_f1.*`, `per_seed_results[*].per_class_accuracy.*`
L1 training metadata	`ml/models/training_metrics.json`	`model_name`, `epochs`, `best_val_acc`, `final_report.*`
L2 auditor accuracy and latency	`ml/eval/results/l2_auditor.json`	`accuracy`, `true_positive_rate`, `false_positive_rate`, `p50_latency_ms`, `p95_latency_ms`, `by_type.*`
L3 PHI leak and channel coverage	`ml/eval/results/l3_phi_leak.json`	`leak_rate`, `overall_detected`, `by_channel.*.detection_rate`, `false_positives`
Layer-by-layer ablation	`ml/eval/results/ablation.json`	`layers[*].config.name`, `asr`, `benign_fpr`, `mean_latency_ms`
L3 ablation deltas	`ml/eval/results/ablation_l3_phi.json`	`with_l3_leaked`, `without_l3_leaked`, `l3_prevention_rate`
L3 micro-latency profile	`ml/eval/results/latency.json`	`l3_scan.p50_ms/p95_ms/p99_ms`, `l3_redact.p50_ms/p95_ms/p99_ms`
Dataset composition and split artifacts	`data/processed/all.jsonl`, `train.jsonl`, `val.jsonl`, `test.jsonl`	Line-level `label` and `source` distribution from processed JSONL files
Runtime thresholds and routing defaults	`configs/aegis.yaml`	`thresholds.auto_proceed`, `thresholds.hold_and_notify`, `thresholds.phi_multiplier`, timeout and mode settings
Runtime enforcement path	`internal/proxy/proxy.go`, `internal/sanitizer/sanitizer.go`, `internal/auditor/auditor.go`	Decision routing, L3 scan/redact behavior, L2 verdict schema

References

Adabara, I., Ssempala, F., Nakayiza, H., Birungi, C., & Mulindwa, D. (2025). Trustworthy agentic AI systems in healthcare: A cross-layer review. *F1000Research*, *14*, 905. doi: 10.12688/f1000research.158204.1

American Hospital Association. (2026). *AHA response to HHS RFI on AI in health care.* https://www.aha.org/lettercomment/2024-09-11-aha-comments-hhs-ai-rfi.

Chen, X., Wang, Y., Yao, Y., Zhang, Z., & Liu, Y. (2025). Emerging cyberattack risks of medical AI agents. *arXiv preprint arXiv:2504.03759*. Retrieved from https://arxiv.org/abs/2504.03759

Dadkhah, S., Carrier, T., Hilal, H., Iqbal, S., & Lashkari, A. H. (2024). CICIoMT2024: A benchmark dataset for multi-protocol security assessment in IoMT. *Internet of Things*, *28*, 101377. doi: 10.1016/j.iot.2024.101377

Dernoncourt, F., Lee, J. Y., Uzuner, Ö., & Szolovits, P. (2017). De-identification of patient notes with recurrent neural networks. *Journal of the American Medical Informatics Association*, *24*(3), 596–606. doi: 10.1093/jamia/ocw156

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *Annals of Statistics*, *7*(1), 1–26. doi: 10.1214/aos/1176344552

Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proc. 16th acm workshop on artificial intelligence and security (aisec '23)* (pp. 79–90). doi: 10.1145/3605764.3623985

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *Proc. icml* (pp. 1321–1330). Retrieved from https://arxiv.org/abs/1706.04599

Jin, D., Pan, E., Oufattole, N., Weng, W.-H., Fang, H., & Szolovits, P. (2021). What disease does this patient have? A large-scale open domain question answering dataset from medical exams. *Applied Sciences*, *11*(14), 6421. doi: 10.3390/app11146421

Jin, Q., Dhingra, B., Liu, Z., Cohen, W. W., & Lu, X. (2019). PubMedQA: A dataset for biomedical research question answering. In *Proc. emnlp-ijcnlp* (pp. 2567–2577). doi: 10.18653/v1/D19-1259

Khan, M. M., Shah, N., Shaikh, N., Thaha, A., Mohamad, S., & Shaikh, S. (2025). Towards secure and trusted AI in healthcare: A systematic review of emerging innovations and ethical challenges. *International Journal of Medical Informatics*, *195*, 105780. doi: 10.1016/j.ijmedinf.2024.105780

Kim, Y., Choi, H., Park, C., Lee, S., Lee, H., & Park, H. W. (2025). Tiered agentic oversight: A hierarchical multi-agent system for safer healthcare AI. *arXiv preprint arXiv:2506.12482*. Retrieved from https://arxiv.org/abs/2506.12482

Kim, Y., Park, C., Jeong, H., Chan, Y. S., Xu, X., McDuff, D., ... Park, H. W. (2024). MDAgents: An adaptive collaboration of LLMs for medical decision-making. In *Advances in neural information processing systems (neurips)*. Retrieved from https://arxiv.org/abs/2404.15155

Liu, Y., Deng, G., Xu, Z., Li, Y., Zheng, Y., Zhang, Y., ... Liu, Y. (2024). Prompt injection attack against LLM-integrated applications. In *Proc. 33rd usenix security symposium.* Retrieved from https://arxiv.org/abs/2306.05499

McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, *12*(2), 153–157. doi: 10.1007/BF02295996

Nagaraja, N., Mishra, S., & Marathe, S. (2025). Cyber threat modeling of an LLM-based healthcare system. In *Proc. 11th int. conf. information systems security and privacy (icissp)*. SCITEPRESS. doi: 10.5220/0013190600003899

Obeid, J. S., Heider, P. M., Weeda, E. R., Matuskowitz, A. J., Carr, C. M., Gagnon, K., ... Meystre, S. M. (2019). Impact of de-identification on clinical text classification using traditional and deep learning classifiers. *Studies in Health Technology and Informatics*, *264*, 283–287. doi: 10.3233/SHTI190228

OWASP Foundation. (2025). *OWASP top 10 for large language model applications and agentic AI security*. https://owasp.org/www-project-top-10-for-large-language-model-applications/.

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*. Retrieved from https://arxiv.org/abs/1910.01108

Schulhoff, S., Pinto, J., Khan, A., Bouchard, L.-F., Si, C., Anati, S., ... Boyd-Graber, J. (2023). Ignore this title and HackAPrompt: Exposing systemic vulnerabilities of LLMs through a global prompt hacking competition. In *Proc. emnlp*. Retrieved from https://arxiv.org/abs/2311.16119

Shah, S. B., Thapa, S., Acharya, A., Rajendran, K., & Goutham, S. (2024). Navigating the risks: A survey of security, privacy, and ethics threats in LLM-based agents. *arXiv preprint arXiv:2411.09523*. Retrieved from https://arxiv.org/abs/2411.09523

Stubbs, A., & Uzuner, Ö. (2015). Annotating longitudinal clinical narratives for de-identification: The 2014 i2b2/UTHealth corpus. *Journal of Biomedical Informatics*, *58*, S20–S29. doi: 10.1016/j.jbi.2015.07.020

Toyer, S., Watkins, O., Mendes, E. A., Svegliato, J., Bailey, L., Wang, T., ... Russell, S. (2024). Tensor trust: Interpretable prompt injection attacks from an online game. In *Proc. iclr.* Retrieved from https://arxiv.org/abs/2311.01011

U.S. Department of Health and Human Services. (2024). *HIPAA privacy rule – safe harbor de-identification, 45 CFR section 164.514(b)(2).* https://www.hhs.gov/hipaa/for-professionals/special-topics/de-identification/.

Wei, A., Haghtalab, N., & Steinhardt, J. (2023). Jailbroken: How does LLM safety training fail? In *Advances in neural information processing systems (neurips).* Retrieved from https://arxiv.org/abs/2307.02483

Yoo, J., Yoon, S. I., Lee, Y., Park, S., & Kim, H. (2025). Vulnerability of large language models to prompt injection attacks in medical advice. *JAMA Network Open*, *8*(12), e2549963. doi: 10.1001/jamanetworkopen.2025.49963

Zhang, Z., Yang, M., Yan, Z., Wang, B., & Goldstein, T. (2024). AgentPoison: Red-teaming LLM agents via poisoning memory or knowledge bases. In *Advances in neural information processing systems (neurips).* Retrieved from https://arxiv.org/abs/2407.12784